APPENDIX S1: Additional Network Testing and Implementation Details

Data Augmentation

The data augmentation used in the present study involved several real-time modifications to the source images at the time of training. Specifically, 50% of all images in a minibatch were randomly modified by means of addition across all pixels using the scalar [−0.1, 0.1], to simulate the effect of random Gaussian noise from differing acquisition parameters, and random affine transformation of the original image, which alters each lymph node slightly by using a rigid transformation, essentially making the same lymph node appear as a unique input to the network. Given the following 2D affine matrix:

$$\begin{bmatrix} s_1 & t_1 & r_1 \\ t_2 & s_2 & r_2 \\ 0 & 0 & 1 \end{bmatrix}$$

the random affine transformation was initialized with random uniform distributions of interval $s_1, s_2 \in [0.8, 1.2]$, $t_1, t_2 \in [-0.3, 0.3]$, and $r_1, r_2 \in [-16, 16]$, where s_1 and s_2 denote shear across x and y axes, respectively, and t1 and t2 denote translation in the x and y axes, respectively. These parameters were confirmed on visual inspection as applying enough of a warp to simulate a different lymph node without making the lymph node appear unrealistic. The choice to apply data augmentation to 50% of the sample images was made to bias the network toward recognition of real data over augmented data.

Network Architecture

The overall network architecture is shown in Fig. 2 and Table S7. The convolutional neural network (CNN) was implemented by a series of 3 × 3 convolutional kernels to maximize computational efficiency while preserving nonlinearity [1]. After an initial standard convolutional layer, a series of residual layers are used in the network. Originally described by He et al. [2] and Szegdy et al. [3], residual neural networks can stabilize gradients during back propagation, leading to improved optimization and facilitating greater network depth. Downsampling of feature map size was implemented by use of a 3 × 3 convolutional kernel with a stride length of two, to decrease size by 75%. All nonlinear functions use the rectified linear unit, which allows training of deep neural networks by stabilizing gradients on backpropagation [4]. In addition, batch normalization was used between the convolutional and rectified linear unit layers, to stabilize training by stabilizing gradients on backpropagation and to prevent covariate shift [5]. After downsampling, the number of feature channels is doubled, reflecting increasing representational complexity and preventing a representation bottleneck. Dropout with a keep probability of 50% was applied to the first fully connected layer to limit overfitting and add stochasticity to the training process [6].

In addition to the customized network described above, several additional network architectures were tested. This includes several 2D neural networks with inputs obtained from the center slice of each spheroid: a ResNet52 (Microsoft) architecture initialized both randomly and with pretrained weights from the Imagenet (Stanford); custom-built networks, initialized from random weights and with varying numbers of convolutional layers based on Inception version 4 architecture; (Google); and a 100-layer network based on a randomly initialized DenseNet architecture (New York University). Performance for the 2D networks was best when weights were initialized randomly across the board. Additional 3D networks were tested by alternating between using inception style layers, residual style layers, and hybrid-wide residual layers. We found that, when greater than 15 hidden layers in 3D residual networks or greater than eight 3D inception style layers were used in this project, overfitting occurred.

Training was implemented using the parameterized Adam optimizer [7, 8], combined with the Nesterov momentum–accelerated gradient described by Dozat [9]. Parameters were initialized to equalize input and output variance utilizing the heuristic described by Glorot and Bengio [10]. L2 regularization was implemented to prevent overfitting of data by limiting the squared magnitude of the kernel weights. Final hyperparameter settings included a learning rate set to 3e − 3, a keep probability for dropout of 50%, a moving average weighted decay of 0.999, and L2 regularization weighting of 1e − 4.

Software code for this study was written in Python programming language (version 3.5, Python Software Foundation) using TensorFlow (version 1.4, Python Software Foundation) open-source software library. Experiments and CNN training were performed on a Linux workstation with the use of a Titan X Pascal graphics processing unit (NVIDIA) with 12 GB on chip memory, an i7 central processing unit (Intel Core), and 32 GB of random-access memory.

References

1. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 1998; 86:2278–2324
2. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. arXiv website. arxiv.org/abs/1512.03385. Submitted December 10, 2015. Accessed October 5, 2018
3. Szegedy C, Liu W, Jia Y, et. al. Going deeper with convolutions. arXiv website. arxiv.org/abs/1409.4842. Submitted September 17, 2014. Accessed October 5, 2018
4. Nair V, Hinton GE. Rectified linear units improve restricted Boltzmann machines. *Proceedings of the 27th International Conference on Machine Learning*. Haifa, Israel: International Machine Learning Society, 2010
5. Ioffe S, Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv website. arxiv.org/abs/1502.03167. Revised September 17, 2015. Accessed October 5, 2018
6. Srivastava N, Hinton G, Krizhevsky A, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* website. jmlr.org/papers/v15/srivastava14a.html. Published 2014. Accessed October 10, 2018
7. Kingma DP, Ba J. Adam: a method for stochastic optimization. arXiv website. arxiv.org/abs/1412.6980. Revised September 17, 2017. Accessed October 5, 2018
8. Nesterov Y. Gradient methods for minimizing composite objective function. *Math Program* 2007; 140
9. Dozat T. Incorporating Nesterov momentum into Adam. Stanford University website. cs229.stanford.edu/proj2015/054_report.pdf. Published 2016. Accessed October 5, 2018
10. Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. MLR Press website. proceedings.mlr.press/v9/glorot10a.html. Published 2010. Accessed October 5, 2018

Fig. S2 — Pie chart of primary tumor histologic subtype in training phase. DLBCL = diffuse large B-cell lymphoma.

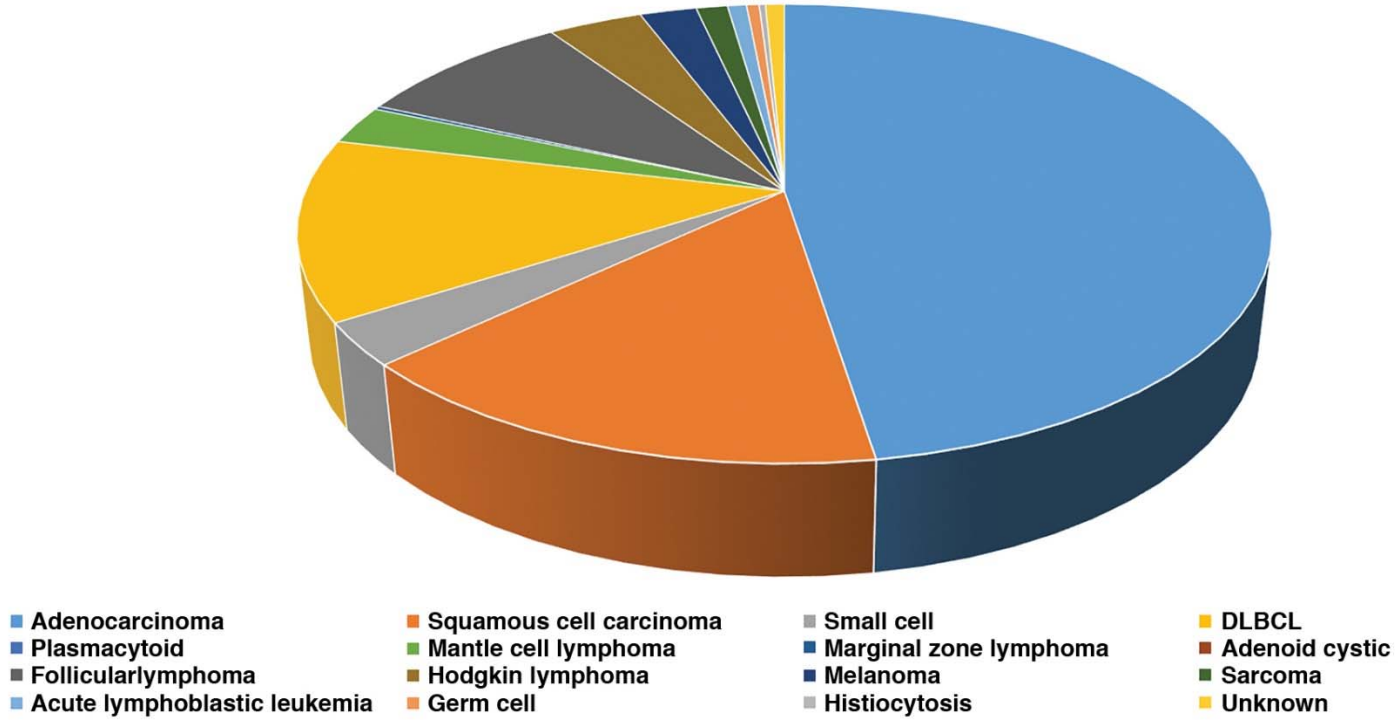

Fig. S3 — Pie chart of primary tumor histologic subtype in testing phase. DLBCL = diffuse large B-cell lymphoma.

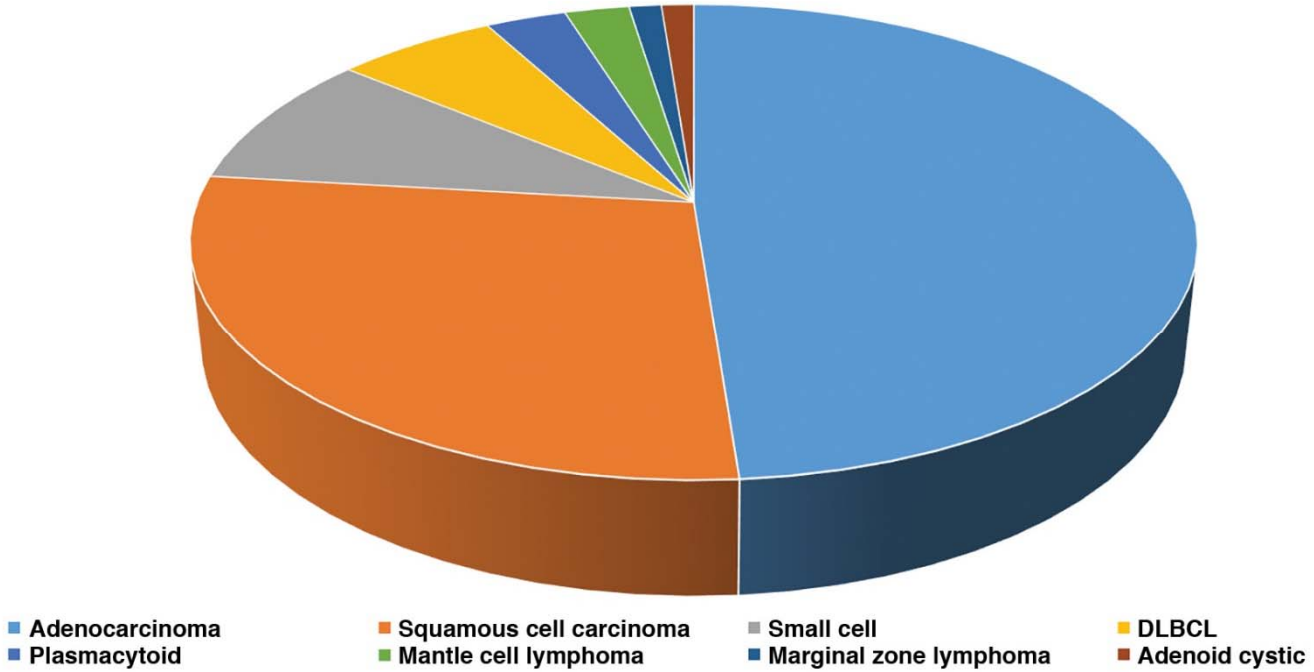

Fig. S4 — Doughnut chart of organ of origin in training phase.

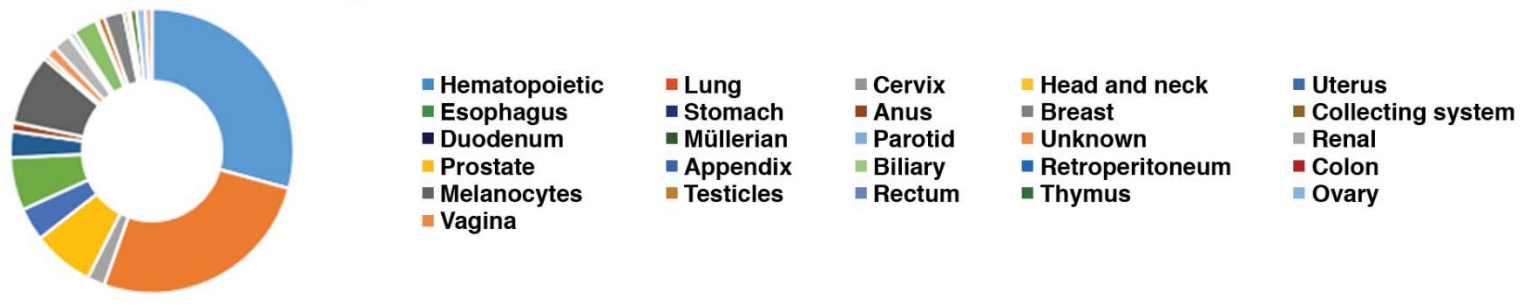

Fig. S5 — Doughnut chart of organ of origin in testing phase.

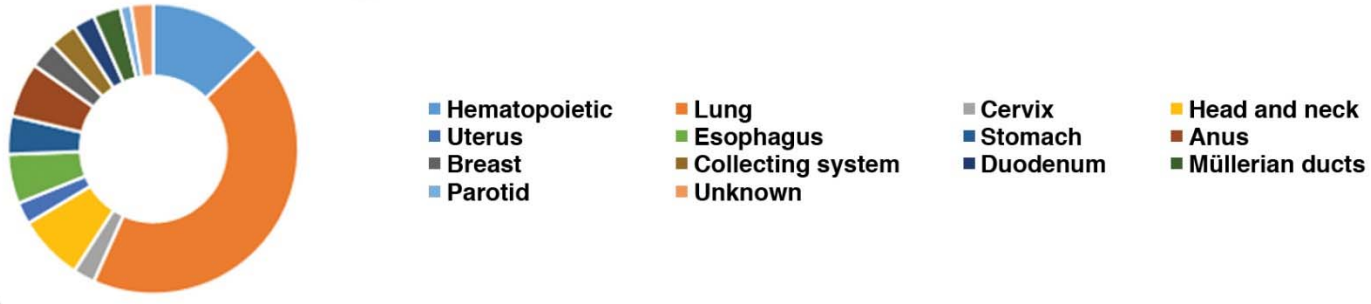

TABLE S6: Histologic Modifiers

Modifier	Training Group	Testing Group
None	371	157
Mucinous	0	2
Signet ring	1	0
Lobular	1	2
Ductal	27	3
Total	400	164